NBSIR 86-3390

Evaluation of L_1 Codes Using Polynomial Approximation Problems

P. D. Domich, K. L. Hoffman, R. H. F. Jackson, P. B. Saunders, D. R. Shier*

Center for Applied Mathematics
National Engineering Laboratory
National Bureau of Standards
Gaithersburg, MD 20899

*This work was performed while D. R. Shier was a member of the Center for Applied Mathematics of NBS. He is currently a professor in the Mathematical Sciences Dept. of Clemson University.

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, Secretary
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, Director

Issued June 1986

NBSIR 86-3390

Evaluation of L_1 Codes Using Polynomial Approximation Problems

P. D. Domich, K. L. Hoffman, R. H. F. Jackson, P. B. Saunders, D. R. Shier*

Center for Applied Mathematics
National Engineering Laboratory
National Bureau of Standards
Gaithersburg, MD 20899

*This work was performed while D. R. Shier was a member of the Center for Applied Mathematics of NBS. He is currently a professor in the Mathematical Sciences Dept. of Clemson University.

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, Secretary
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, Director

Issued June 1986

ABSTRACT

This paper presents the methodology and results of a computational experiment which compares the performance of four computer codes which determine the best discrete L_1 approximation to a continuous nonlinear function. The experiment utilizes 320 test problems created by a test problem generator. Several performance measures describe solution quality as well as computational effort.

Key Words: Algorithm testing; approximation; computational experiment; least absolute deviation; performance measures; polynomial approximation.

I. INTRODUCTION

This paper is part of a series of papers dealing with computational experimentation in mathematical programming, which emphasizes sound, statistically based methodology for comparing codes in this area. As the specific object of our experimentation (and, therefore, a vehicle for developing a sound methodology), we chose to study codes that solve problems of least absolute value curve fitting. Estimation in the L_1 norm provides a useful adjunct to least squares estimation [12,20], especially when errors derive from long-tailed distributions [15,16].

In earlier work [14], we studied four codes, representative of a range of solution techniques, using a statistically based experiment design and pseudo-randomly generated test problems (representative of general L_1 data-fitting problems). It was discovered that such codes could be clearly ranked with respect to computational effort (CPU time), but that very little difference among the codes could be observed with respect to certain "quality-of-results" performance indicators: accuracy, consistency, and correctness.

Therefore, another experiment was designed using polynomial approximation test problems and the same four L_1 codes used in the earlier work. Polynomial approximation problems are frequently encountered in practice [7,19,25], and are known to admit some of the interesting properties of ill-conditioning [11] which tend to differentiate among codes with respect to the quality of results.

Moreover, prior to executing the first comparison experiment, we tested the codes on a battery of hand-picked test problems obtained from the authors, the literature and code users [22]. Each of the codes yielded incorrect information on at least one of the hand-picked problems, with polynomial approximation problems being especially troublesome.

This paper will present an experiment for testing codes on a class of polynomial approximation problems, and discuss the results of this test effort. Directions for future research in both code development and code-comparison methodology will be presented in the final section of this report.

II. BACKGROUND

Given n sets of observations (y_i, x_i) i=1, . . . , n, the discrete L_1 polynomial approximation problem is that of determining values $\beta_0, \ldots, \beta_m$ which minimize

$$0\ (\beta_0, \ldots, \beta_m) = \sum_{i=1}^{n} \left| y_i - \sum_{j=0}^{m} \beta_j x_i^j \right| \quad . \qquad (1)$$

In other words, for n observations on a dependent variable y and an explanatory variable x, we wish to determine a polynomial of degree m which best fits these data, in the sense of minimizing the sum of the absolute values of the residuals

$$e_i = y_i - \sum_{j=0}^{m} \beta_j x_i^j \quad .$$

A vector $\beta^* = (\beta_0^*, \ldots, \beta_m^*)$ which yields the minimum value 0* of $0\ (\beta_0, \ldots, \beta_m)$ is termed a solution vector, with optimum objective function value 0*. Unlike the case for L_1 polynomial approximation over a continuous interval [25], page 38, the solution vector for a discrete L_1 polynomial approximation problem need not be unique.

Discrete L_1 polynomial approximation problems can be formulated and solved as linear programming problems by the stratagem of introducing variables $\alpha_i = |e_i|$:

$$\text{minimize} \quad \sum_{i=1}^{n} \alpha_i$$

$$\text{subject to} \quad \alpha_i + \sum_{j=0}^{m} \beta_j x_i^j \geq y_i \quad , \qquad i = 1, \ldots, n \quad ,$$

$$\alpha_i - \sum_{j=0}^{m} \beta_j x_i^j \geq -y_i \quad , \qquad i = 1, \ldots, n \quad . \tag{2}$$

It can be demonstrated [2,4,13] that a solution vector β^* can always be found that interpolates at least m+1 of the data points; that is, at least m+1 residuals e_i are equal to zero at the solution. Degeneracy is said to occur when more than m+1 residuals have value zero at a solution β^*.

Certain algorithms for solving general discrete L_1 approximation problems make use of the above "interpolation" result. Namely, these algorithms restrict attention to basic solutions (where precisely m+1 residuals equal zero), and move in a systematic way from one basic solution to an improved basic solution. Any such collection of m+1 zero-valued residuals corresponds, then, to the specification of m+1 active constraints: values selected from i = 1, . . . , n, where both inequalities of (2) hold with equality.

Four general-purpose discrete L_1 approximation FORTRAN codes were selected for study and evaluation in this paper. Three of these codes are based on linear programming reformulations of the L_1 approximation problem:

(I) A double-precision dual revised simplex code by Abdelmalek [1],

(II) A primal revised simplex code with a partial sort procedure by Armstrong [3],

(III) A primal simplex code with full tableau and limited use of double-precision by Barrodale and Roberts [5].

While codes II and III are essentially primal simplex codes, they do make use of certain dual properties in passing over several simplex vertices in a single iteration.

In addition, a restricted gradient method for minimizing piecewise differentiable functions is embodied in a fourth code:

(IV) A descent code by Bartels [6], incorporating a form of QR decomposition.

A May 1977 version of each code was obtained from the authors. Tolerance settings were fixed at those levels suggested by the authors as appropriate for our computer (UNIVAC 1108 with 36-bit words). In code IV, a random perturbation is internally applied to the problem data when near-degeneracy is suspected.

Unfortunately, when using this code as a subroutine to solve a series of identical problems, the actual perturbation applied varies from problem to problem. In order to ensure that the same (perturbed) problem is solved in successive executions using the same input problem, the pseudo-random number generator provided in Code IV was always initialized prior to execution. Other than these changes (tolerance settings, and pseudo-random number initialization), the codes were implanted exactly as provided by the authors. Certain general characteristics of the various codes are indicated in Table I.

Table I

General Code Characteristics

	Code I	Code II	Code III	Code IV
Number of Statements	270	186	223	1609
Required* Storage	$2np+2p^2+7n+6p+1151$	$np+p^2+6n+4p+824$	$np+5p+5n+723$	$np+1.5p^2+5n+6.5p+4496$
Restrictions	None	Full Rank	None	$m>1$
Additional Features	None	None	None	Allows constraints, variable starting points, selective printing

*Defined in terms of n = number of observations, and p = m+1 = number of estimated parameters.

In our previous study of these codes [14], two types of test problems were considered: (1) a collection of 27 hand-picked problems [22], devised with a specified structure in mind or arising from an actual application; and (2) pseudo-randomly generated problems for general discrete L_1 approximation [17]. While for certain classes of pseudo-randomly generated problems, all four codes "correctly" solved each generated problem, this was not the case for many of the 27 hand-picked problems. Since a number of these "difficult" problems were in fact forms of discrete polynomial approximation in the L_1 norm, it appeared useful to study systematically the performance of the four codes on a wider class of polynomial approximation problems. To this end, test problem generators have been developed [9] to create a range of polynomial approximation problems representative of situations encountered in practice.

One test problem generator (POLY1) models the situation in which a given nonlinear function $f(x)$ is to be approximated in the L_1 norm by a polynomial of specified degree, over a discrete set. Accordingly, the observed values y_i in (2) are modeled as arising from the given function, via $y_i = f(x_i)$. Controllable features of this generator include: the choice of function $f(x)$, the interval I of approximation, the distribution and number of observation points $x_i \varepsilon I$, and the degree of the approximating polynomial. For this particular generator, the actual solution vector β^* cannot be specified in advance. However, in the second generator (POLY2) for another class of polynomial approximation problems, the solution vector can be so specified; the current experiments have concentrated on the use of POLY1, and thus attention will subsequently be confined to results obtained using this first generator.

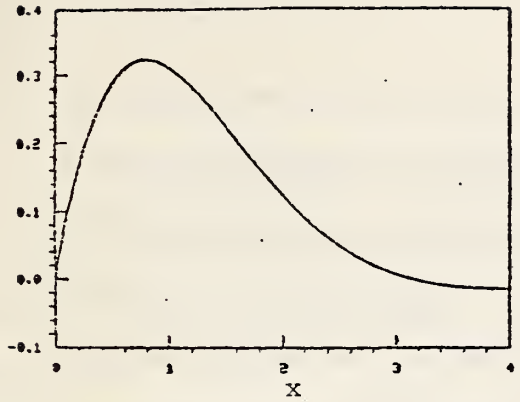
With POLY1 one can study discrete polynomial approximation of a variety of functional forms, defined on certain intervals. The eight functions f(x) employed in the present code evaluations are listed in Table II, together with their associated domains. These functions, and their intervals of definition, were chosen in order to represent a useful variety of shapes likely to be encountered and (through the choice of interval) a range of "ill-conditioning" for the problem matrix $X = (x_i^j)$. Figure 1 graphs the light functions studied in this paper.

TABLE II

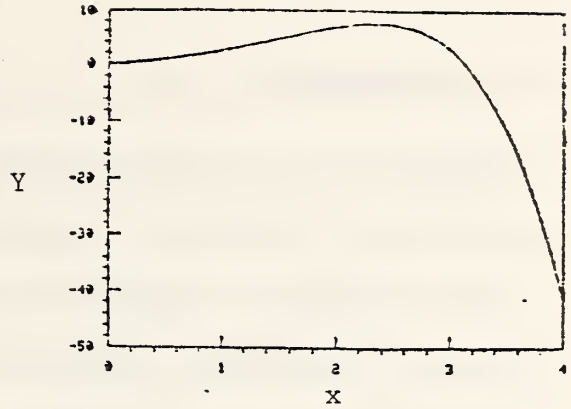
Functions and Their Intervals of Approximation

	Function f(x)	Interval I
1	$e^{-x} \sin x$	[0,4]
2	$e^{x} \sin x$	[0,4]
3	$e^{x} \sin x$	[1,7]
4	$e^{2x}/2x$	[.05,1]
5	$75x/[1+(7.5x)^2]$	[0,2]
6	$10 \times e^{-.5x}$	[0,4]
7	$1/[1+(x-2.5)^4]$	[0,5]
8	$x^{1/3}$	[-1,1]

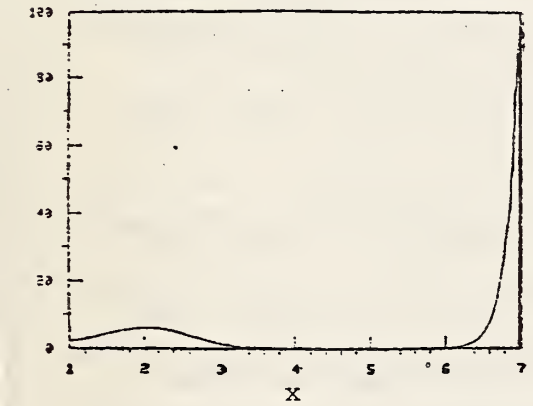
1. $Y=e^{-x}\sin(x)$



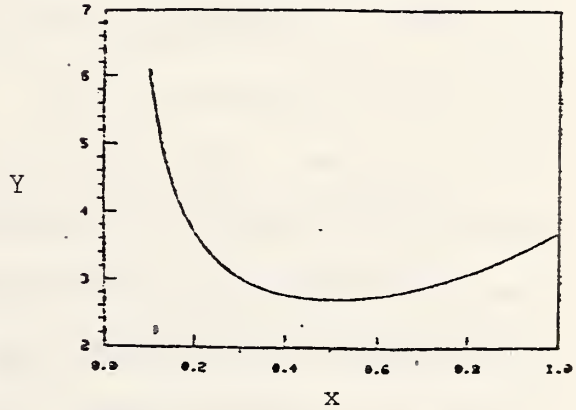
2. $Y=e^{x}\sin(x)$



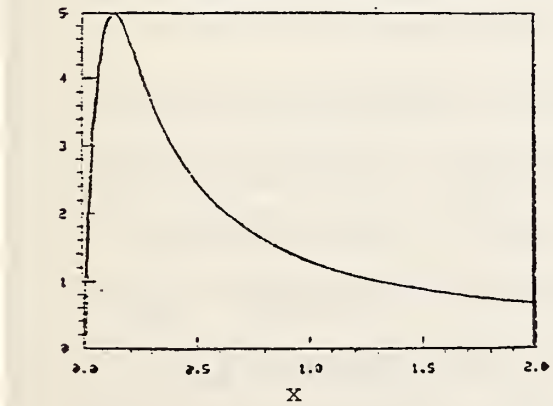
3. $Y=e^{x\sin(x)}$



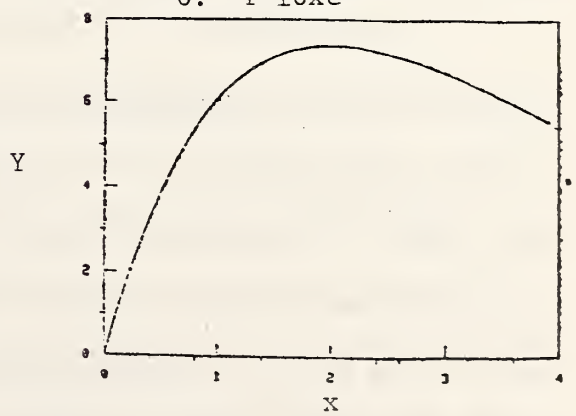
4. $Y=e^{2x}/2x$



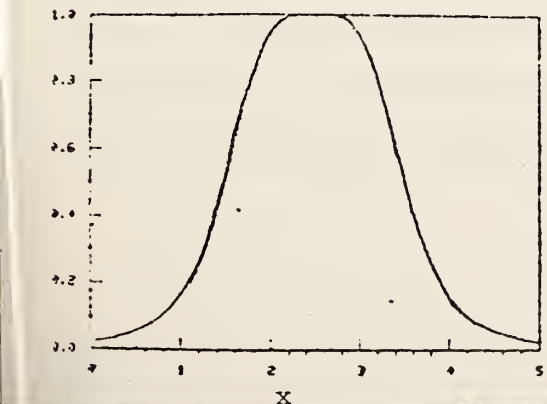
5. $Y=75x[1+(7-5x)^{2}]$



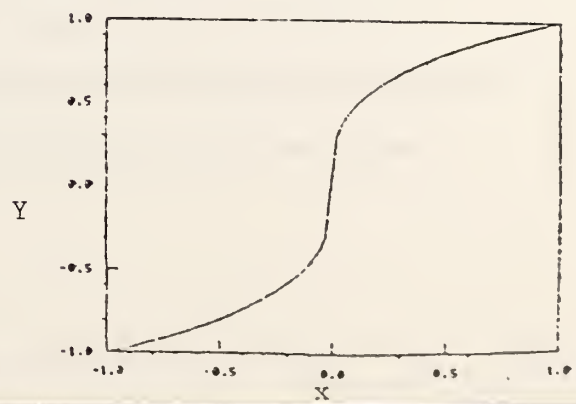
6. $Y=10xe^{-.5x}$



7. $Y=1/[1+(x-2.5)^{4}]$



8. $Y=CBRT(x)$



III. METHODOLOGY

The approach taken in comparing the four L_1 codes consisted of designing a computational experiment to discern the effects of certain controlled variables by examining a number of performance indicators. This section of the paper will describe the details of this experiment.

The first step in designing the experiment involved the identification, specification, and quantification of the controlled variables which characterize the problems generated by POLY1. These controlled variables fall into three main categories: those describing the function to be approximated, those describing the spacing of the observations within the interval over which the function is approximated, and those describing the size of the problem. For the experiment described here, the functions used were the eight functions described in Table II together with their associated intervals of approximation. Observations were chosen to be evenly spaced within the interval.

The size of each approximation problem was specified by the number of observations n and the degree m of the approximating polynomial. For the experiment the number of observations assumed the values 50, 100, 200, 400, and 800 and the approximating polynomial had degree varying from 3 to 10 inclusive. Thus each of the four L_1 codes was exercised on a total of 320 problems (8 functions, 8 degrees, 5 sets of equally spaced observations).

The second part of the experiment design involved identifying and quantifying various indicators of code performance. Performance indicators can be grouped naturally into two general categories: those related to computational effort and those related to the quality of the solution produced. The indicators in both categories must be judged relative to the particular computer system and computer environment in which the experiment is conducted. All of the results reported in this paper were obtained on the UNIVAC 1108 at the National Bureau of Standards under the Exec 8 operating system, Version 33.R3, with 36 bits for each single precision computer word.

Perhaps the most frequently used performance indicator is Central Processing Unit (CPU) time. In this experiment CPU time was chosen as the appropriate indicator of computational effort inasmuch as the four codes were sufficiently different for iteration counts to be meaningless. Quantification of this indicator was obtained by accessing an internal clock from a system subroutine which references an appropriate elapsed CPU time entry (in milliseconds) in the run accounting table. Previous results in timing experiments on our computer [13] indicated several sources of variability in the process which confound code comparison results. First, the inherent variability associated with clock and accounting table update was found to be on the order of 2 or 3 milliseconds. A second possible contributor to variability occurs because the execution time depends on the locations in core memory of instructions and the data upon which those instructions operate. To reduce this effect, the dimensions of the arrays used by all four codes are fixed and do not vary with problem size. A third source of timing variability is the computing environment: the number and types of other jobs being run during the same time

period. Previous results [14] indicated that for any environment other than a totally dedicated computer, one run of a code on a problem would not suffice to determine true CPU time on that problem. Thus, all of the timing results reported here were obtained under totally dedicated conditions.

The second category of performance indicators studied relates to the quality of the solution obtained. This category (which is not always considered) proved to be critically important in the present experiment, whereas in our previous experiment [14], it did not play a decisive role in distinguishing among the codes. The evaluation of solution quality requires either knowledge of the "true" solution or the ability to check a given solution for optimality. As mentioned in Section 2, the polynomial approximation problems produced by POLY1 cannot be generated with solutions known a priori. Thus, it was necessary to devise a procedure for checking solution quality. Accordingly, we developed a double precision computer subroutine which, given the problem data and specified set of active constraints, checks the Kuhn-Tucker conditions for optimality [4,17]. If satisfied, it then performs a double-precision reinversion of the optimal basis matrix to obtain accurate optimal values for the objective function and the solution vector. These solution values are accepted as the "true" values for purposes of comparison with the results produced by the L_1 codes.

Four different performance indicators were chosen to represent solution quality: correctness, accuracy, accuracy after reinversion, and consistency. All four indicators involve comparisons between "true" and code-derived active

constraints, solution vectors or objective function values. Several quantifications were used in this study. For comparing objective function values, the experiment calculated the number of digits of agreement, the absolute difference, and the relative difference. For comparing solution vectors, the experiment examined average number of digits of agreement, absolute Euclidean distance, and normalized Euclidean distance (where the two vectors β^*_{code} and β^*_{true} were viewed as points in $(m+1)$-dimensional space). Precise definitions of these indicators and their quantification are deferred to the next section.

The experiment design reported here thus consists of the specification of problems to be solved by each of the four L_1 codes and the identification of the indicators upon which code performance will be evaluated and compared. The next section presents the results of executing this experiment design.

IV. RESULTS

Each of the four codes was tested using all 320 test problems. Elapsed CPU time and four measures for quality of results (correctness, accuracy, accuracy after reinversion, and consistency) were obtained.

CORRECTNESS

Correctness (did the code actually arrive at the true solution?) is a measure of performance of interest to all users. We defined correctness as a binary (YES/NO) measure. If the active constraints defining the final solution produced by an L_1 code satisfied the Kuhn-Tucker conditions to given tolerances, it is called "correct". In our analyses, we noticed that some codes produced an objective function value which agreed with the true solution to eight significant digits but whose final simplex vertex was not an optimal vertex. Since comparisons using more than eight significant digits are inappropriate using single-precision arithmetic on the UNIVAC 1108, whenever the objective function produced by a single precision code agreed with that of the true solution to eight digits, we also considered the solution "correct". For ten of the 320 problems, no code produced the correct solution judged according to Kuhn-Tucker conditions. These problems were eliminated from our subsequent analysis since the "true" solution could not be ascertained. Analyses of the results using the correctness measure are presented in Figures 2 - 4. There is no table for Code I which obtained the correct solution for every problem analyzed (310 of the 320). Code II produced incorrect results--regardless of function type--for almost all problems of degree greater than 5 or for

Figure 2.

CODE II – CORRECT SOLUTIONS

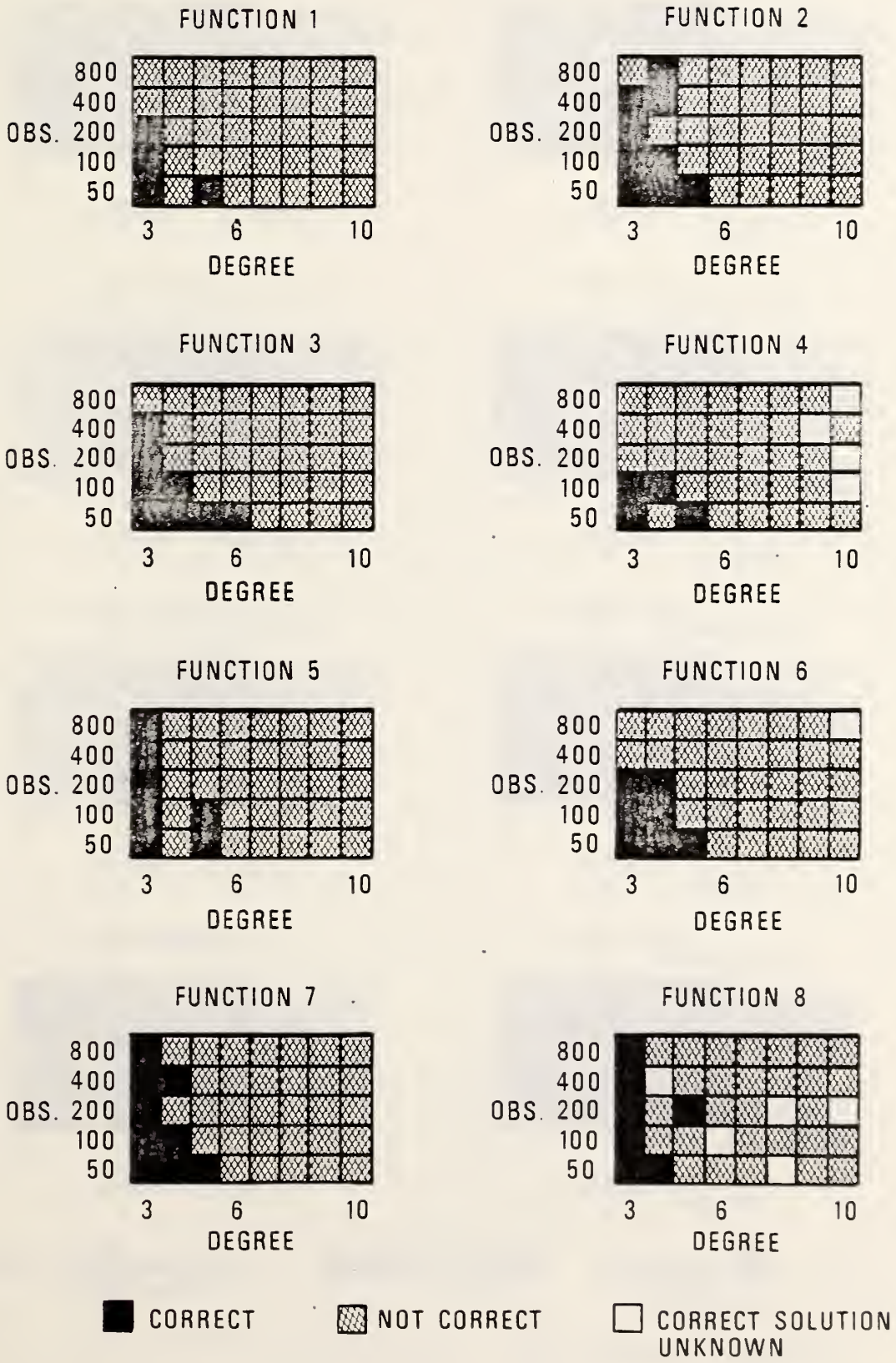


Figure 3.

CODE III–CORRECT SOLUTIONS

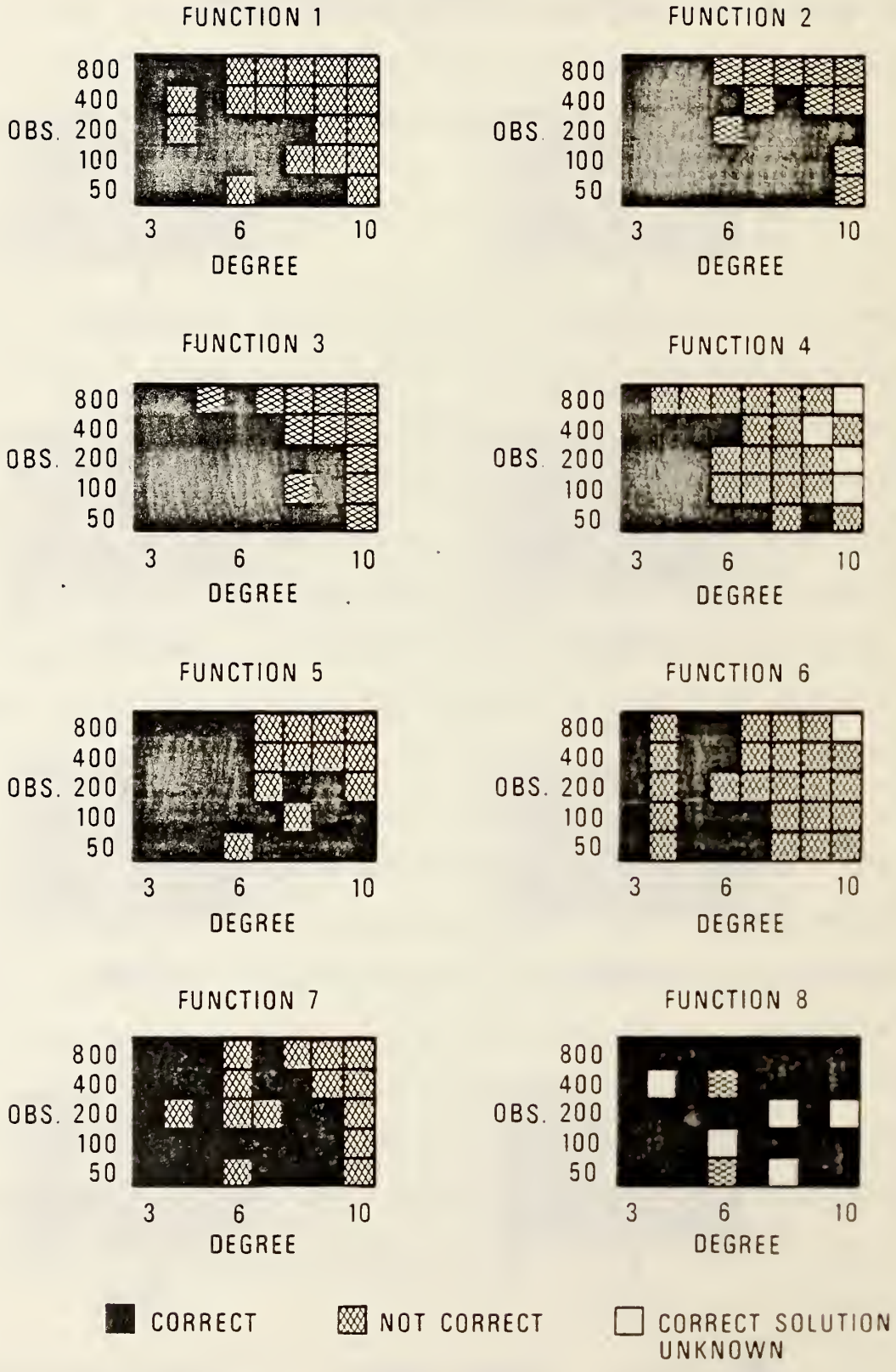
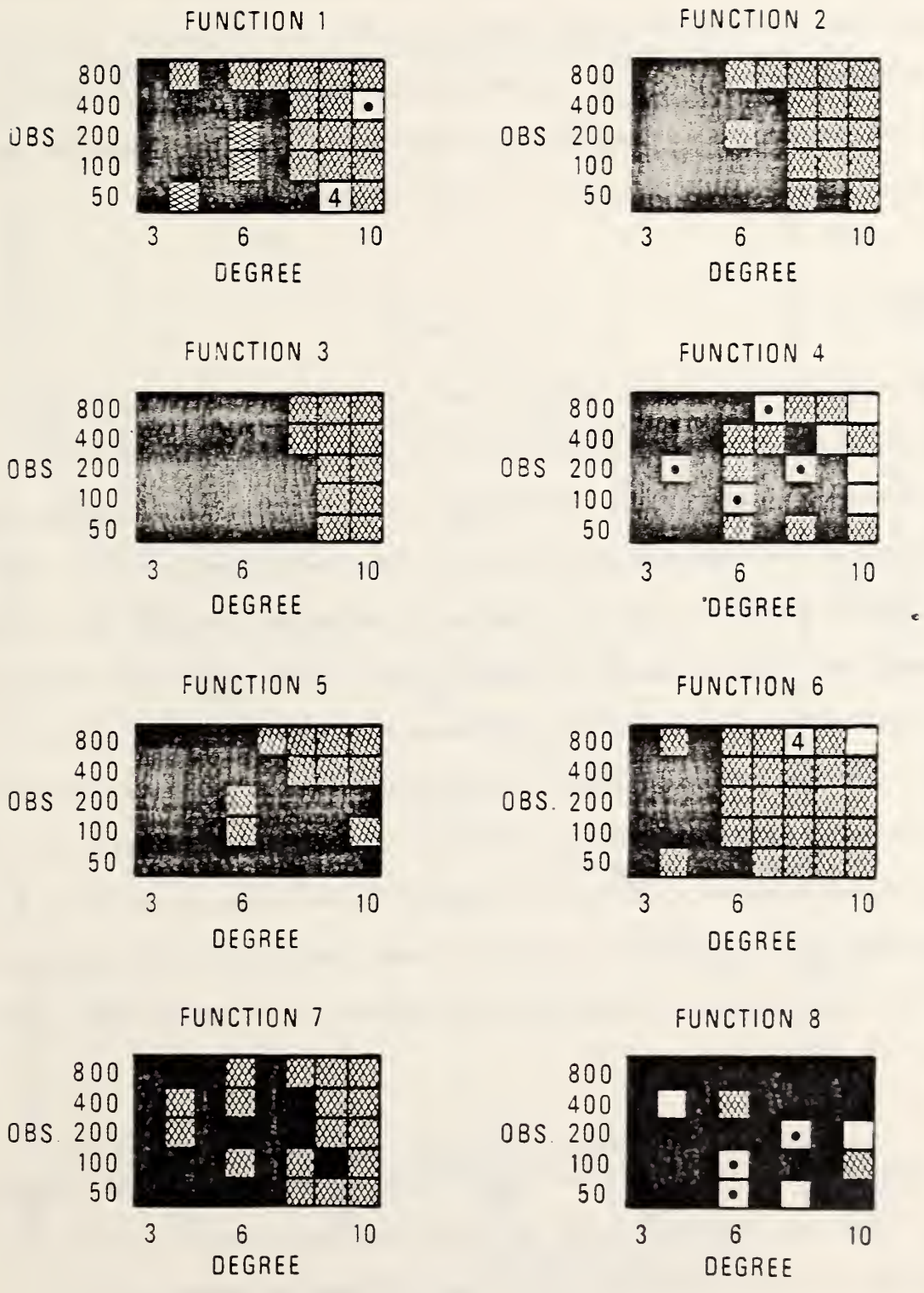


Figure 4.

CODE IV–CORRECT SOLUTIONS



problems with greater than 100 observations. Codes III and IV were less frequently correct than Code I but more often correct than Code II. Although one can conclude that both of these codes are likely to be judged "not correct" for problems having both high degree and many observations, consistent patterns did not emerge to delineate the function-type or problem sizes which are most likely to cause difficulty to either of these codes.

ACCURACY

Having established that all codes do not yield the correct solution to all problems, the next natural question is how close to correctness are the answers provided by the codes. Accordingly, the performance indicator, accuracy, compares the true solution with that obtained by the L_1 code. Figures 5 - 7 present for each code the fraction of solutions produced with errors in the objective function values no greater than a given magnitude, where error is quantified by three measures: the absolute difference, the relative difference, and the number of digits of agreement between the true objective function value and that produced by the L_1 code. The test results for all 310 test problems produces a definitive ranking of the codes, with Code I being the superior code, followed by Code III, then Code IV and finally Code II. Note that this ranking is invariant with respect to the measurement quantification.

Three measures were developed for quantifying the accuracy of the solution vector. The true solution vector and that produced by each L_1 code can be considered as points in $(m+1)$-space. The Euclidean distance and the

Figure 5.

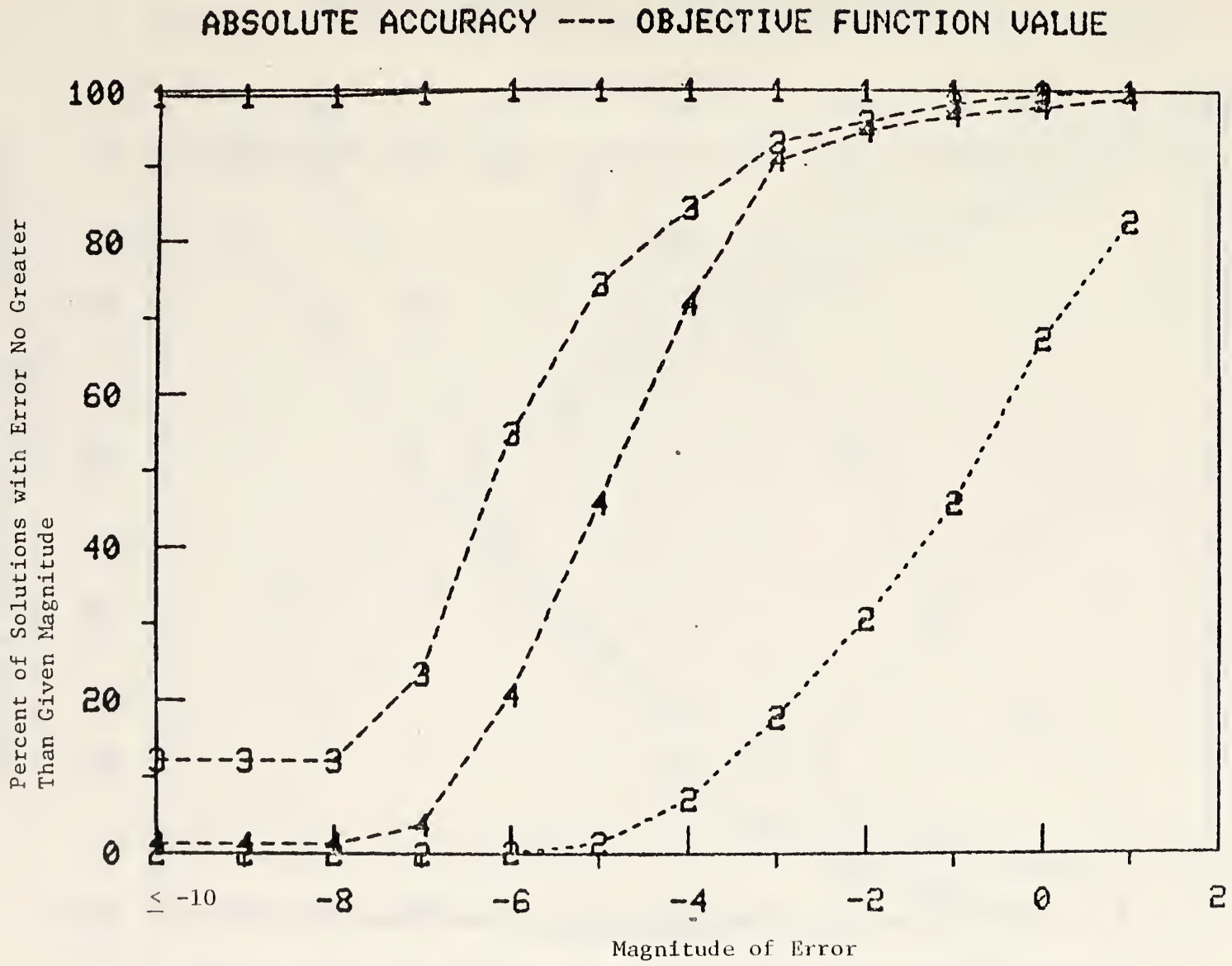
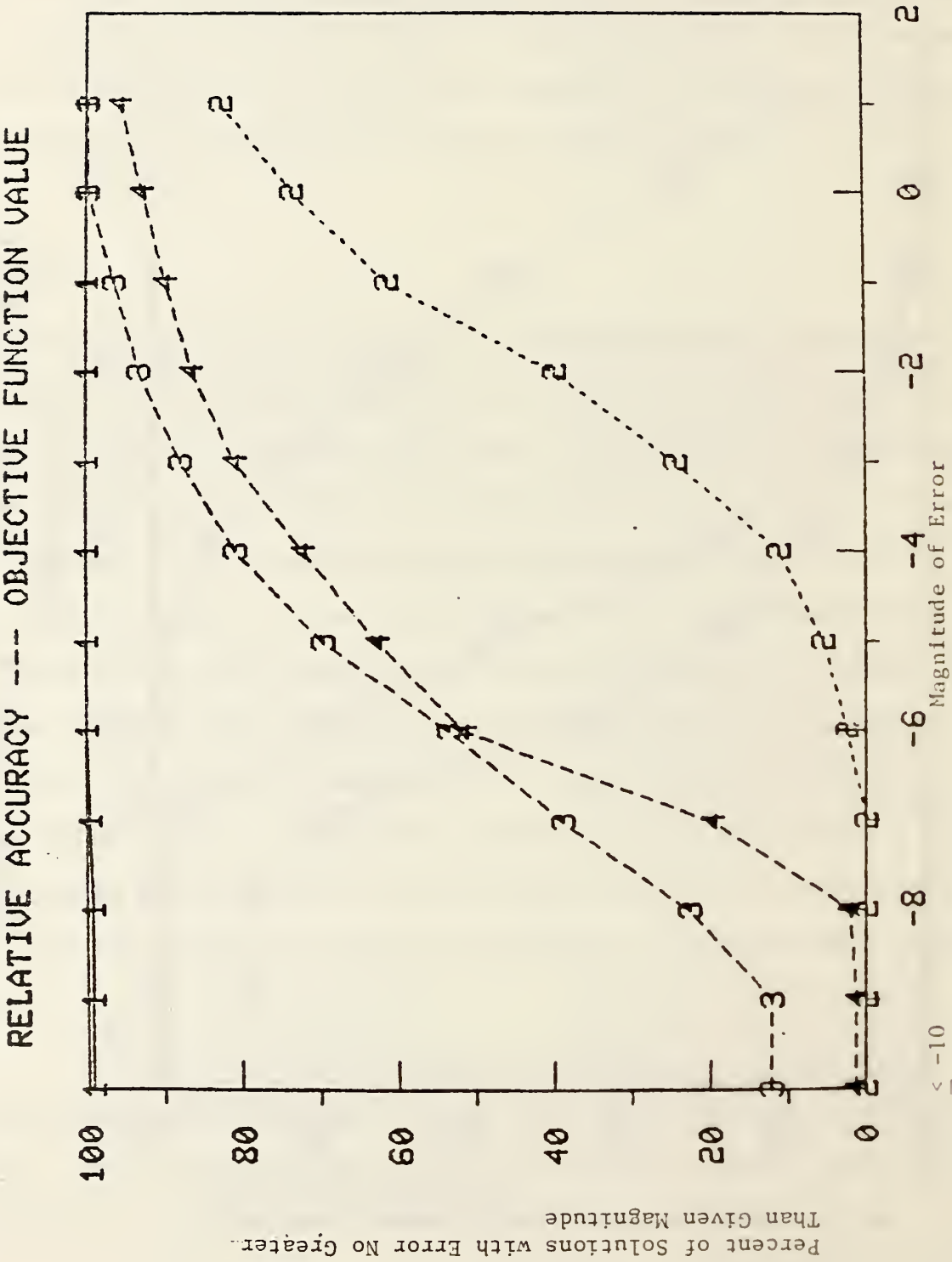

ABSOLUTE ACCURACY --- OBJECTIVE FUNCTION VALUE
Percent of Solutions with Error No Greater Than Given Magnitude
100
80
60
40
20
0
≤ -10
-8
-6
-4
-2
0
2
Magnitude of Error
SOLID = CODE I
DOTTED = CODE II
DASH = CODE III
DASH" = CODE IV

Figure 6.


RELATIVE ACCURACY --- OBJECTIVE FUNCTION VALUE
Percent of Solutions with Error No Greater Than Given Magnitude
100
80
60
40
20
0
≤ -10
-8
-6
-4
-2
0
2
Magnitude of Error
SOLID = CODE I
DASH = CODE III
DOTTED = CODE II
DASH" = CODE IV

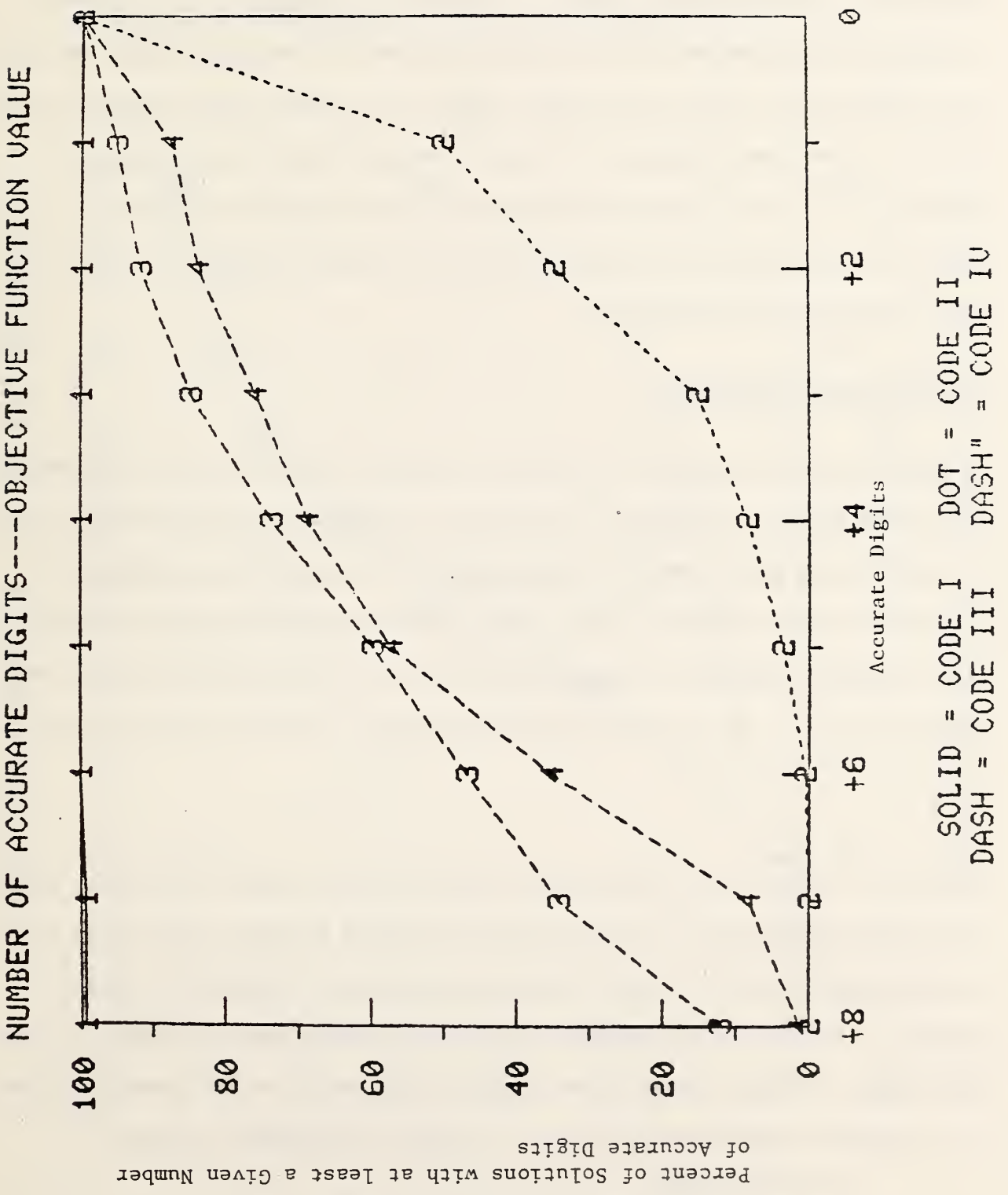

NUMBER OF ACCURATE DIGITS---OBJECTIVE FUNCTION VALUE
Percent of Solutions with at least a Given Number of Accurate Digits
100
80
60
40
20
0
+8
+6
+4
+2
0
Accurate Digits
SOLID = CODE I DOT = CODE II
DASH = CODE III DASH" = CODE IV

Figure 7.

normalized Euclidean distance between the points are then two measures of the accuracy of the solution vectors produced by the L_1 codes. A third measure is obtained by calculating the number of digits of agreement (0 to 8) between corresponding components of the solution vectors and averaging these over the m+1 components to obtain the average number of accurate digits in the solution vector. These three measures of solution vector accuracy are presented in Figures 8 - 10. Note that the ranking of the four codes is essentially the same as that obtained for objective function accuracy except that Codes III and IV are more evenly matched.

ACCURACY AFTER REINVERSION

In performing the analyses of these test results, we noticed that frequently a code would arrive at a "correct" vertex but the objective function value and/or the solution vector would be "inaccurate". We, therefore, performed a double-precision inversion of the final basis matrix and compared the objective function value and solution vector obtained after inversion to the true solution values. We call this performance measure "accuracy after reinversion".

We note that for Code IV the results here are based on only 279 (rather than 310) problems because 31 of the solutions produced by Code IV had fewer than m+1 active constraints, thereby rendering reinversion impossible. Figure 11 presents the results of examining the relative differences in objective function values. Similar results were obtained for the other two quantifications, i. e. absolute difference and number of digits of agreement. Figure 12

Figure 8.

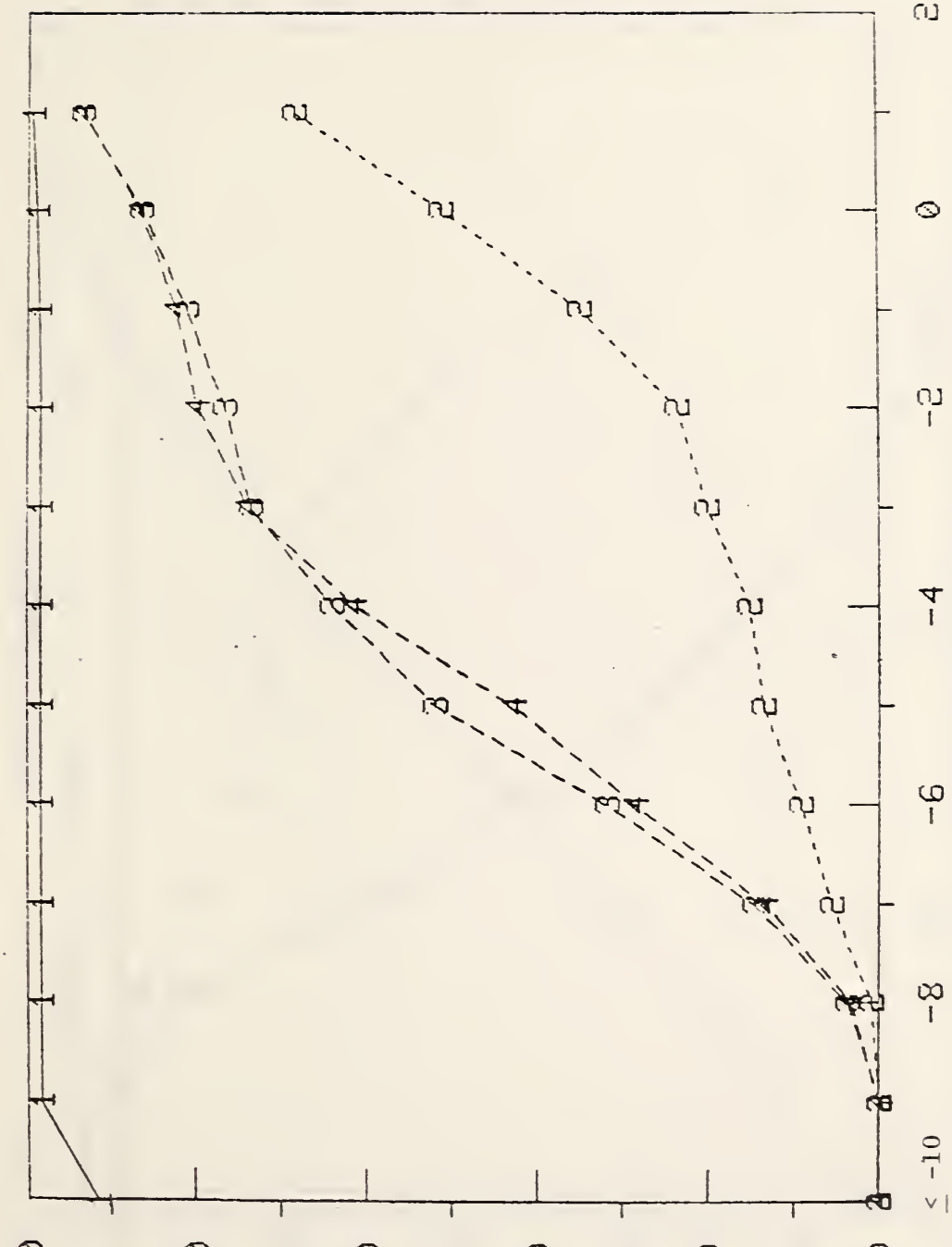

EUCLIDEAN DISTANCE --- SOLUTION VECTOR
Percent of Solutions with Error No Greater Than Given Magnitude
100
80
60
40
20
0
≤ -10
-8
-6
-4
-2
0
2
Magnitude of Error
SOLID = CODE I
DOT = CODE II
DASH = CODE III
DASH" = CODE IV

Figure 9.

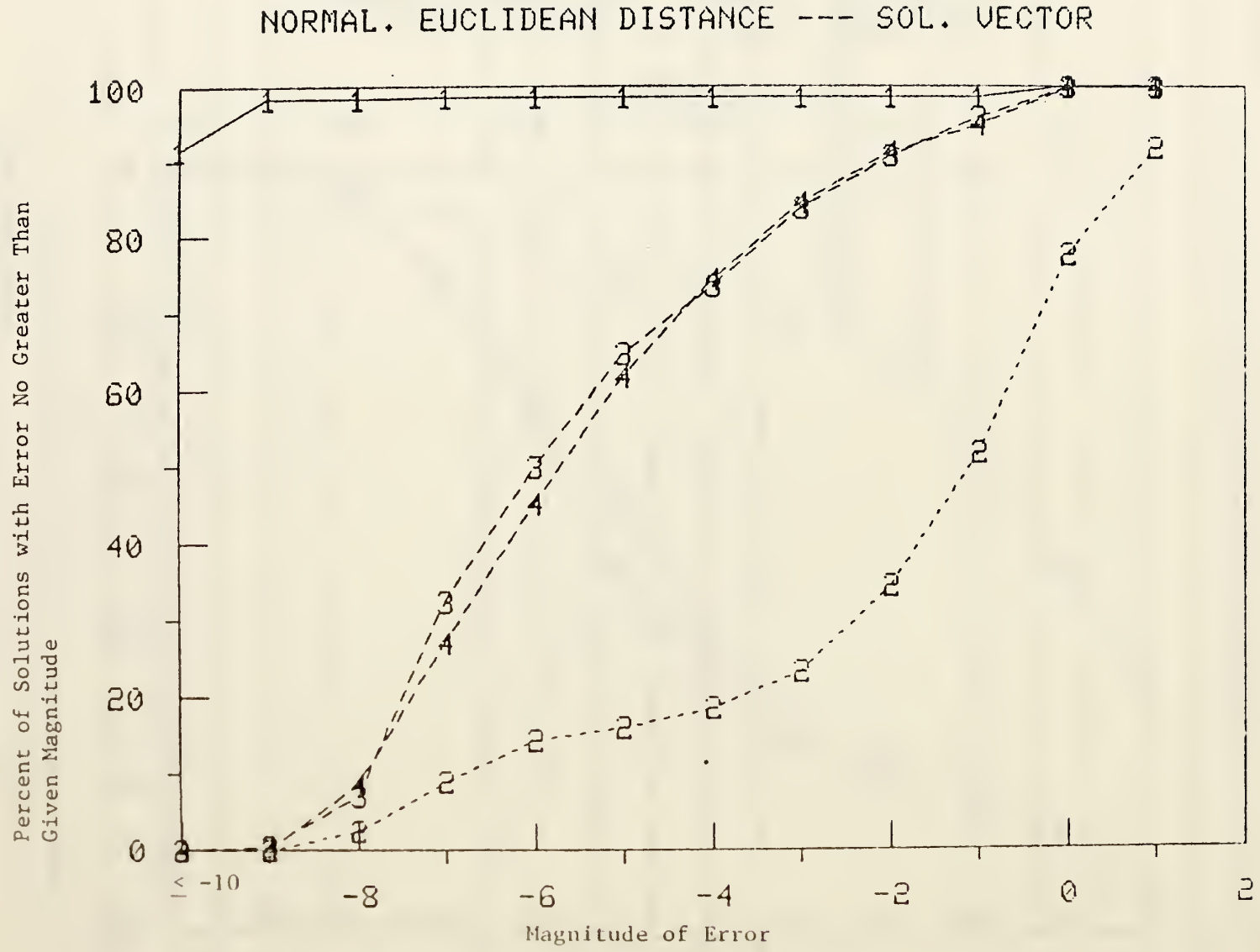
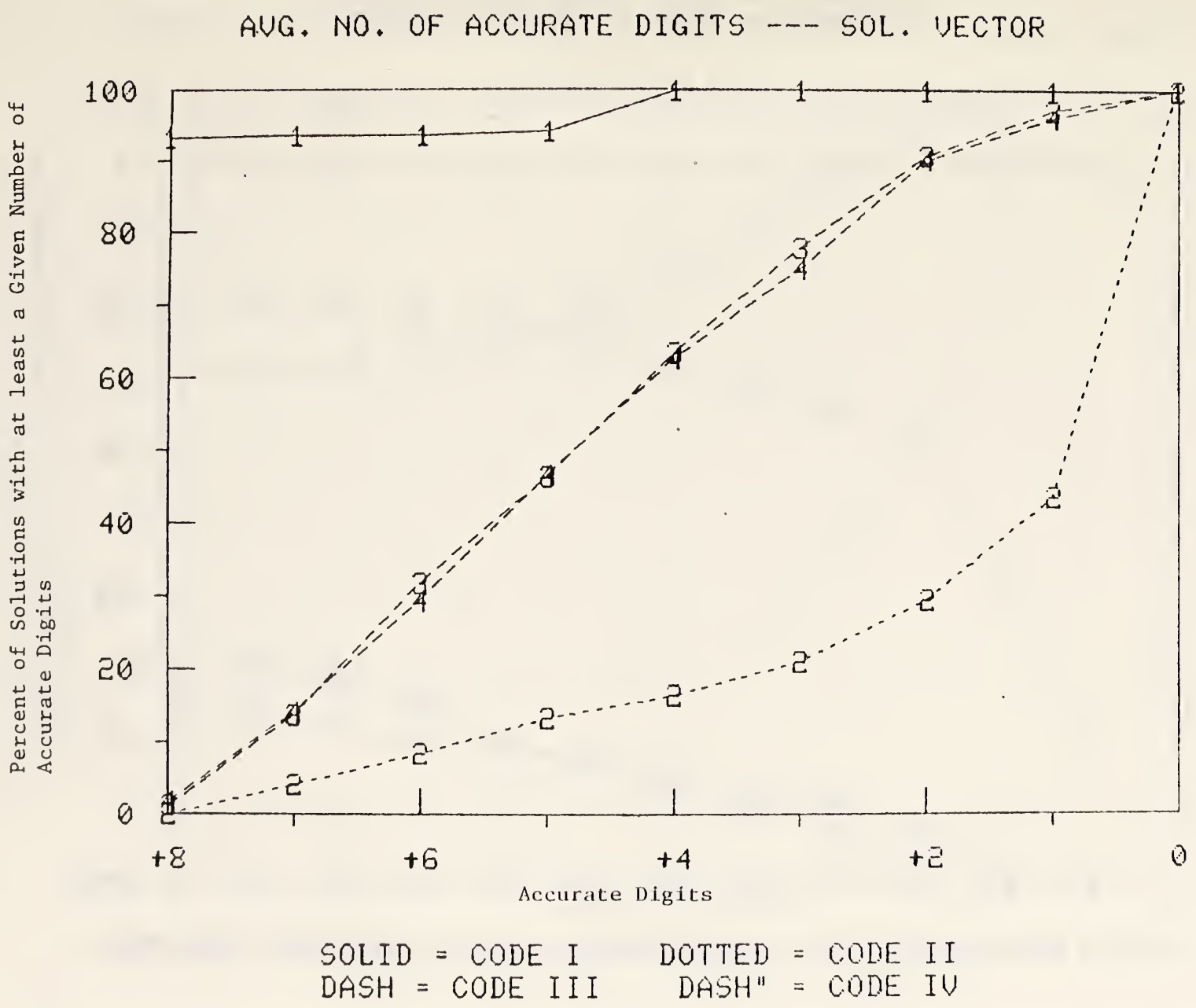

NORMAL. EUCLIDEAN DISTANCE --- SOL. VECTOR
Percent of Solutions with Error No Greater Than Given Magnitude
100
80
60
40
20
0
≤ -10
-8
-6
-4
-2
0
Magnitude of Error
SOLID = CODE I
DOTTED = CODE II
DASH = CODE III
DASH" = CODE IV

Figure 10.


AVG. NO. OF ACCURATE DIGITS --- SOL. VECTOR
Percent of Solutions with at least a Given Number of Accurate Digits
100
80
60
40
20
0
+8
+6
+4
+2
0
Accurate Digits
SOLID = CODE I
DOTTED = CODE II
DASH = CODE III
DASH" = CODE IV

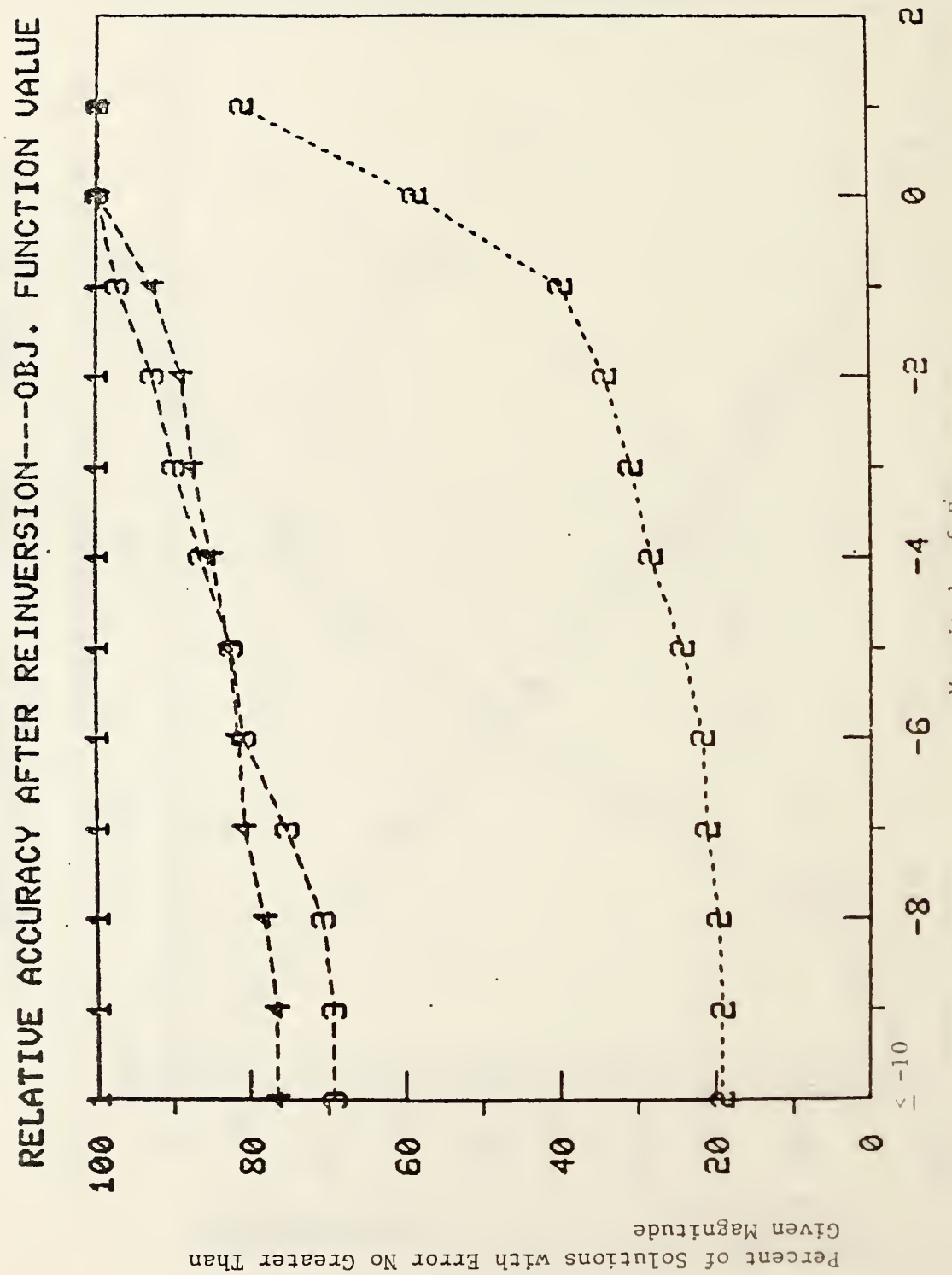
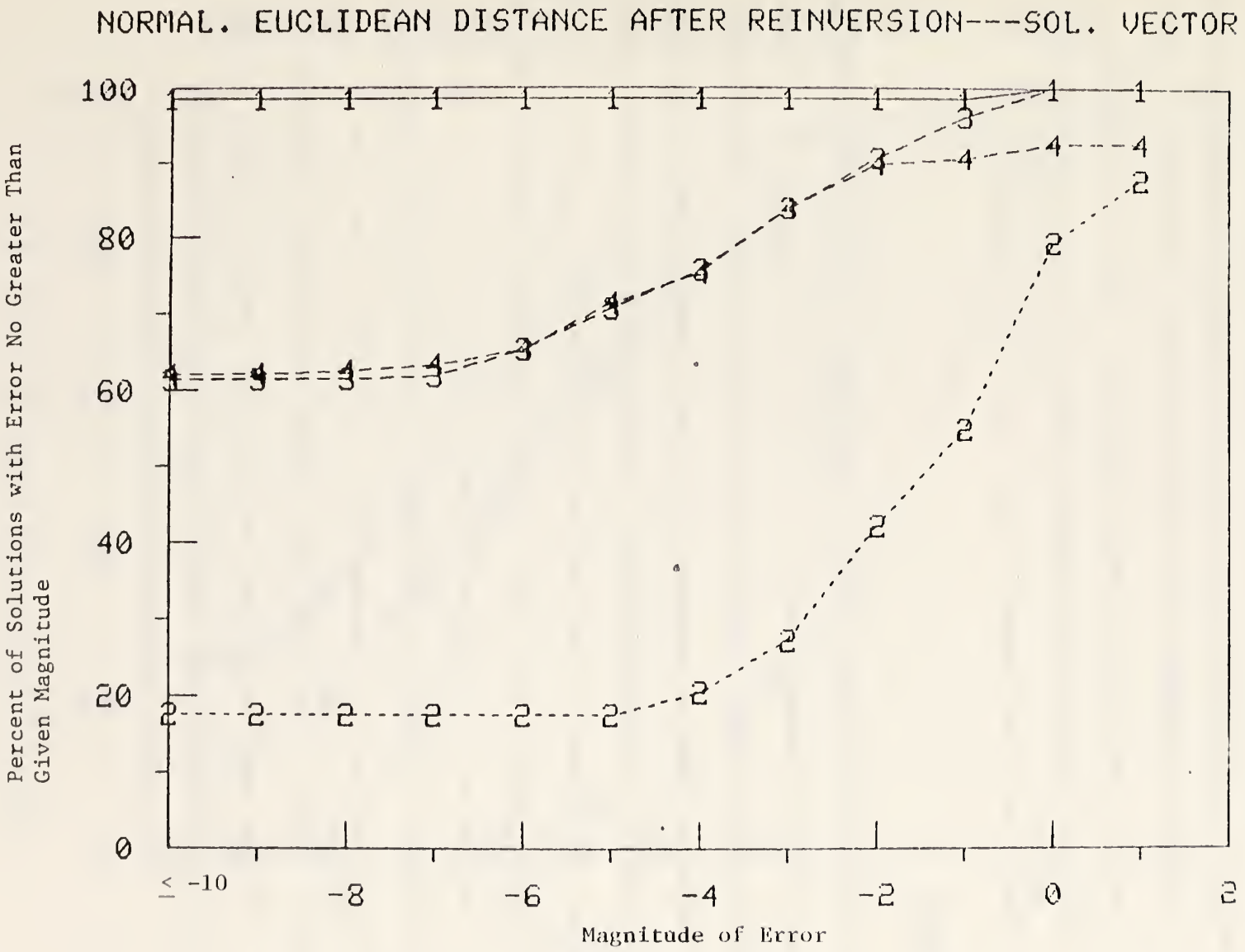

RELATIVE ACCURACY AFTER REINVERSION---OBJ. FUNCTION VALUE
Percent of Solutions with Error No Greater Than Given Magnitude
100
80
60
40
20
0
≤ -10
-8
-6
-4
-2
0
2
Magnitude of Error
SOLID = CODE I DOT = CODE II
DASH = CODE III DASH" = CODE IV

Figure 11.

Figure 12

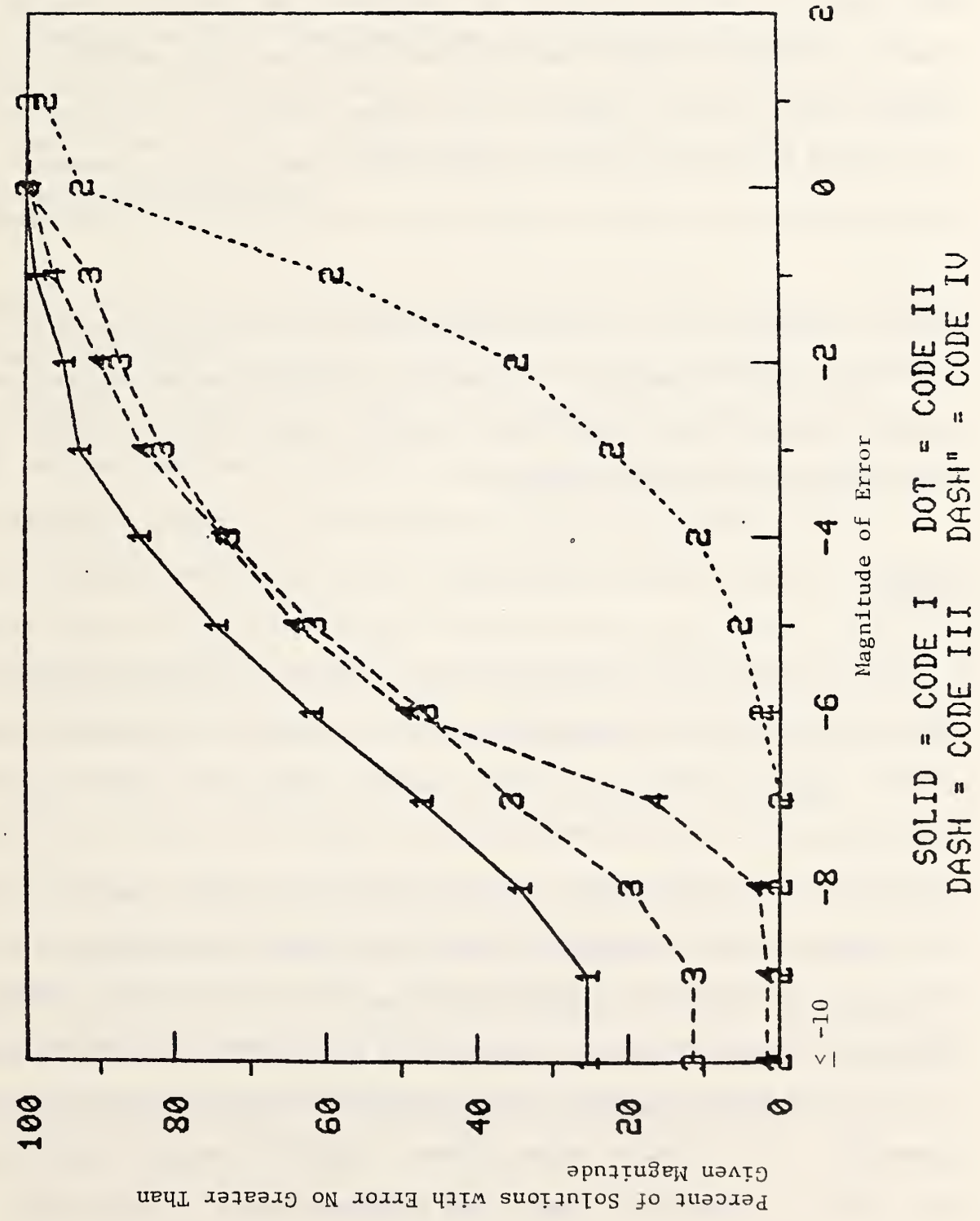

NORMAL. EUCLIDEAN DISTANCE AFTER REINVERSION---SOL. VECTOR
Percent of Solutions with Error No Greater Than Given Magnitude
100
80
60
40
20
0
≤ -10
-8
-6
-4
-2
0
2
Magnitude of Error
SOLID = CODE I
DOTTED = CODE II
DASH = CODE III
DASH" = CODE IV

illustrates the accuracy in the solution vector after reinversion as quantified by the normalized Euclidean distance between the two points. Again, similar results were obtained for the absolute Euclidean distance and the average number of digits of agreement. Note that the accuracy of codes II, III, and IV have improved substantially through reinversion. Thus, if one were to use any of the single precision codes tested in the study, it is clear that an accurate (double precision) inversion of the final basis is desirable.

CONSISTENCY

A fourth indicator of solution quality, consistency, provides some information about the effects of round-off error. It refers to the internal consistency between the solution vector and the objective function value produced by the code. The solution vector produced by the code is used with the given problem data to calculate, in double-precision, the residuals and an objective function value. Consistency is then measured by comparing this objective function value with that given by the L_1 code. As can be seen from Figure 13, approximately 40 percent of the solutions produced by Codes II, III, and IV yielded inconsistencies in the solution information such that the relative differences in objective function value exceeded 10^{-5}. The ranking of the codes for this performance measure is not unlike that produced for the other performance measures. Note that even the "correct" and "accurate" results of Code I are not consistent, an indication that round-off error is affecting the results.

Figure 13.


RELATIVE CONSISTENCY OBJECTIVE FUNCTION VALUE
Percent of Solutions with Error No Greater Than Given Magnitude
100
80
60
40
20
0
≤ -10
-8
-6
-4
-2
0
2
Magnitude of Error
SOLID = CODE I DOT = CODE II
DASH = CODE III DASH" = CODE IV

In summary, one observes that Code I--a double-precision code--was superior to the other three single-precision codes with respect to all measures of all four quality of results criteria. It should also be noted that Code III, which is a single-precision code using one double-precision variable to accumulate inner products, generally outperforms the other two single-precision codes. This inexpensive and core-conserving device should be considered in all single-precision codes where numerical difficulties may be anticipated.

Finally, although two of the three single-precision codes had a switch to indicate that they terminated due to numerical difficulties and although there were many instances when significant round-off errors existed, only rarely did a code terminate with this warning.

TIMING

As stated in Section III, CPU time for runs made on a dedicated computer was used as an indicator of computational effort. In a previous comparison experiment [14] with the four L_1 codes, solution quality was "perfect" and the only mechanism for distinguishing among the codes was in terms of CPU time. The results of that experiment indicated that Code II always required the least amount of time, followed by Codes III, I, and IV, generally in that order. For the experiment described herein, the solution quality became of significant importance and, as a result, CPU time became less important since it can be argued that solution time is irrelevant when solution quality is unacceptable. It can also be argued, however, that CPU time is always important since (1) it is perhaps the single most important factor in calculating

computer run costs under many computer accounting systems, (2) the codes rarely produced warnings that solution quality was perhaps suspect, and (3) one does not know in advance of code execution just what to expect in terms of the solution time and quality. Thus, CPU times were analyzed for three of the four codes studied in this experiment. Code II was omitted from the analysis of CPU time because of the very large number of problems for which solution quality was deemed unacceptable.

CPU times for Codes I, III, and IV were analyzed for all 320 problems (includes the 10 problems for which no code found a correct solution) with one general exception. There were a total of 18 problems for which Code IV apparently entered some sort of infinite loop, stopping the solution process only by exceeding a maximum of 1000 iterations without determining a result which the code considered a solution. These eight problems required a total of slightly over 13 minutes of computer time and were not included in the results shown for Code IV. Because Code IV generally requires more CPU time than the other codes, the omission of these exceptionally large times does not significantly alter the ranking of the codes with respect to CPU time.

Figures 14, 15, and 16 illustrate various results with respect to CPU time. Figure 14 illustrates the average CPU time in milliseconds as it relates to the number of observations. The results here indicate that with the possible exception of a very slight reversal of the order between Codes I and II at 50 observations, Code I requires the least CPU time followed by Codes III and IV.

Figure 14.

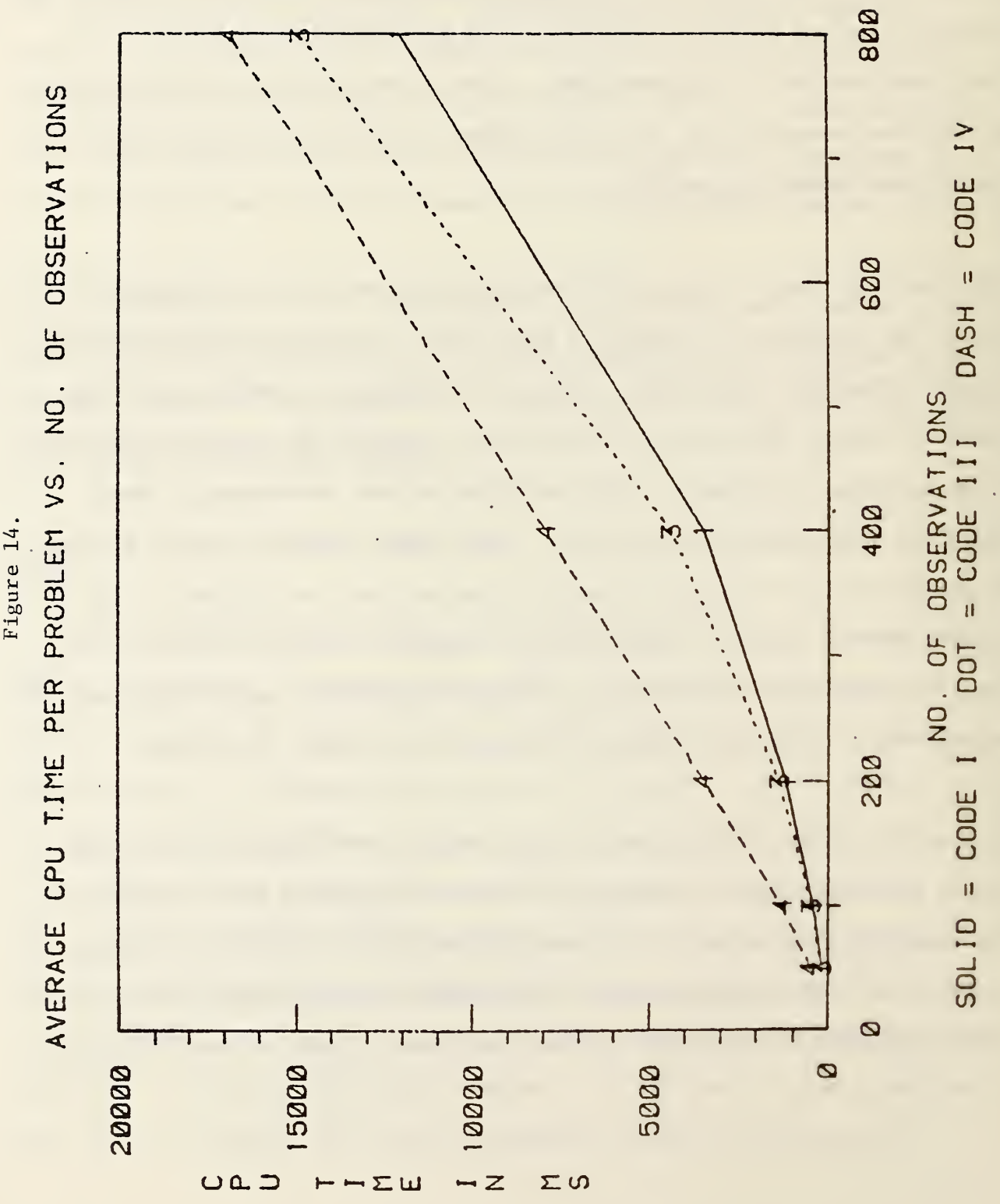

AVERAGE CPU TIME PER PROBLEM VS. NO. OF OBSERVATIONS
CPU TIME IN MS
20000
15000
10000
5000
0
0
200
400
600
800
NO. OF OBSERVATIONS
SOLID = CODE I DOT = CODE III DASH = CODE IV

Figure 15

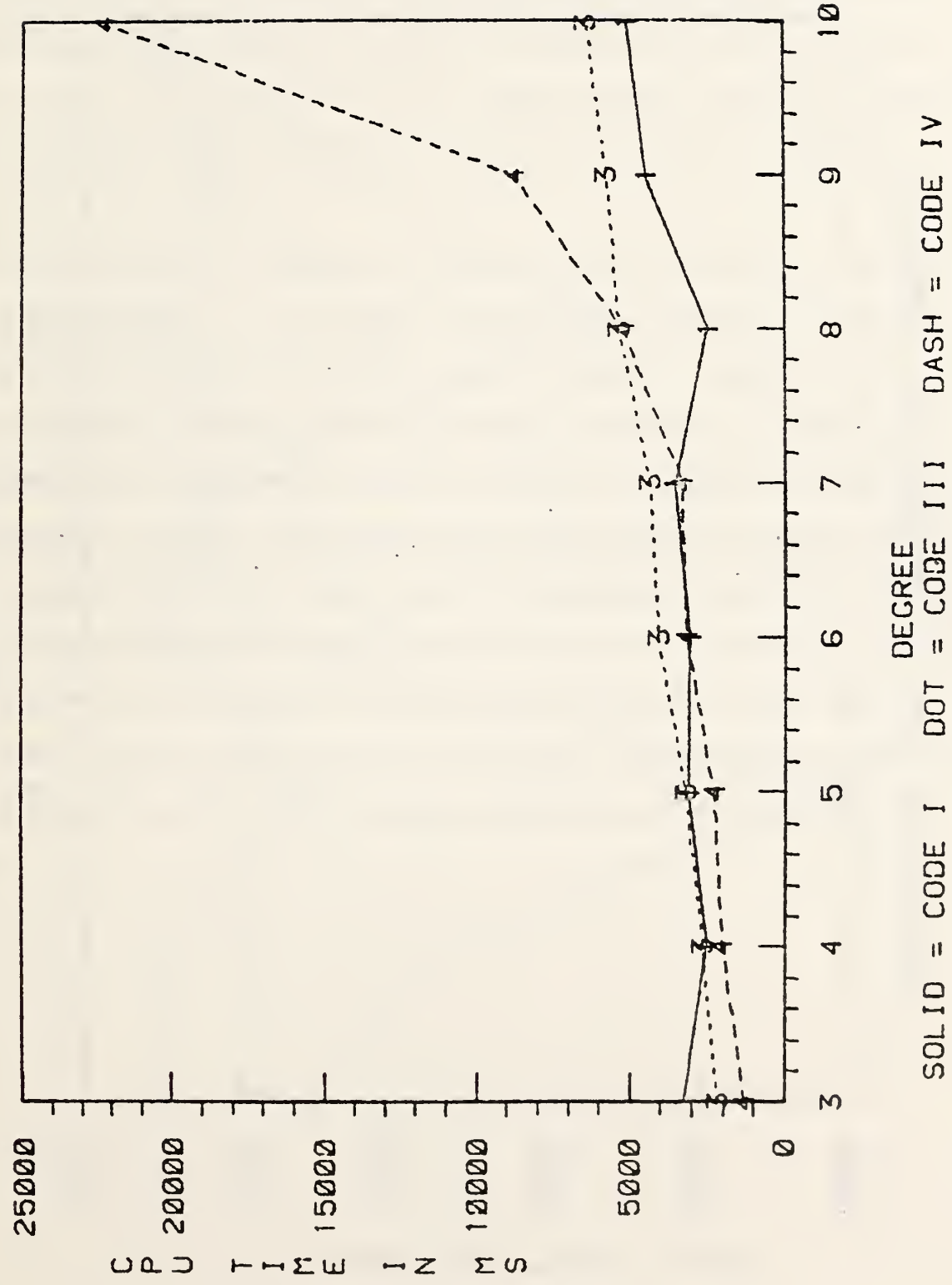
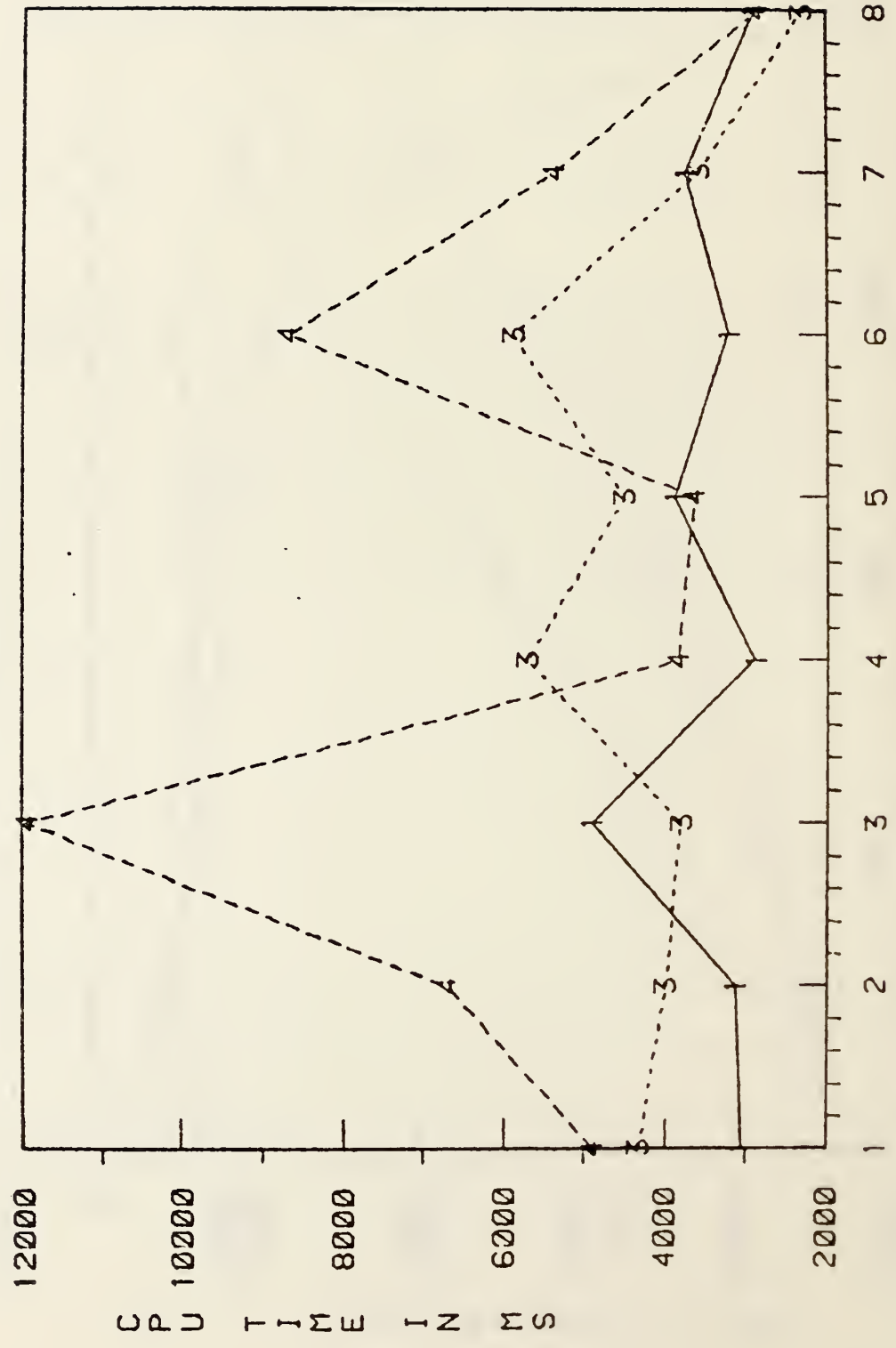

AVERAGE CPU TIME PER PROBLEM VS. DEGREE
CPU TIME IN MS
25000
20000
15000
10000
5000
0
3
4
5
6
7
8
9
10
DEGREE
SOLID = CODE I
DOT = CODE III
DASH = CODE IV

Figure 16.


AVERAGE CPU TIME PER PROBLEM VS. FUNCTION
CPU TIME IN MS
12000
10000
8000
6000
4000
2000
1
2
3
4
5
6
7
8
FUNTION NUMBER
SOLID = CODE I
DOT = CODE III
DASH = CODE IV

In Figure 15, the results are not quite so definitive. This figure illustrates average CPU time as it relates to the degree of the approximating polynomial. Here, Code IV performs well for polynomials of low degree, but uses noticeably more time for degrees 9 and 10. In contrast, Codes I and III perform fairly well and consistently for all degrees studied. Except for degree 3, the average CPU time for Code I was always less than that for Code III.

Figure 16 illustrates the relationship between average CPU time and the function to be approximated. This figure illustrates quite clearly the need for comparing code performance over a wide class of problems. Consider, for example, the difference of opinion that would result if one were to examine code performance only on function 3 or only on function 5 versus the examination of all 8 functions. Indeed, to the extent that longer CPU time is indicative of greater problem "difficulty", the results here demonstrate conclusively that problems which are the most difficult for one code are not necessarily the most difficult (and at times are even the least difficult) for some other code. Thus, valid statements regarding comparative code performance cannot be made solely on the basis of computational experience with a small number of problems.

V. OTHER RESULTS

A preliminary investigation was also made of the effect of "ill-conditioning" on the nature of test problems and the behavior of L_1 codes on those problems. For the purpose of this investigation, the 1-condition number (or H-condition number) of the optimal basis matrix B was chosen as a measure of problem difficulty [10,18]. An estimate k(B) of this condition number can be efficiently computed [10], with the interpretation that calculation of β^*, using $B\beta^* = y_B$, involves the loss of approximately $\log_{10}k(B)$ significant figures of accuracy.

It was found that the condition number estimate k(B) did not vary appreciably with n (the number of observations), but it was strongly affected by m (the degree of the approximating polynomial). In fact, for each unit increment in m, k(B) grows by approximately one order of magnitude. Furthermore, for fixed n and m, the condition estimate varies by function - with function 3 giving the largest and function 8 the smallest such estimates. This occurence is not unexpected, in that function 3 is approximated over a fairly broad interval [0,7], while function 8 is approximated over a fairly narrow interval [-1,1]. More generally, k(B) was observed to increase with the increasing magnitude of the interval endpoints. Such results are consistent with the notion that the successive powers $1, x_i, \ldots, x_i^m$ comprising matrix B become more widely divergent in magnitude as m increases or $|x_i|$ increases.

In order to assess how the accuracy of the L_1 codes changed in response to problem characteristics, the average number of agreeing digits (AAD) between the true solution vector β^*_{true} and that provided by the code β^*_{code} was

computed. Agreement was invariably perfect for Code I, and so subsequent analysis focused on Codes II - IV. For these latter codes, AAD did not vary in a discernable way with n, but generally exhibited a decrease with increasing values of m. While this behavior of AAD parallels that found for k(B), there was no consistent relation between this measure of accuracy and the condition estimate for a given degree m. In other words, very little additional information on code accuracy is conveyed by knowing the condition estimate, in addition to the value of m.

REGULARLY SPACED POINTS OF EXACT FIT

The test problems used within this study were generated by varying the form of the function to be approximated, the degree of the approximating polynomial, and the number of observation points. The observation points were positioned evenly spaced over the interval of interest.

When the interpolation points defining the best L_1 solution to each problem were tabulated--keeping the degree of the polynomial and the number of observaton points fixed but allowing the function-type to vary--a pattern emerged which indicated that these interpolation points would be regularly spaced within the interval regardless of the functional form chosen.

An understanding of approximation theory presents a partial explanation for this phenomenon. A theorem in this field [21, pp. 71-73] states that the best L_1 approximating polynomial of degree m-1 (or less) to x^m over [-1,1] has its

interpolation points at the values $x_k = \frac{\cos k\pi}{m+1}$ $(k = 1, \ldots, m)$. These values x_k are the roots of a Chebyshev polynomial of the second kind.[1] This theorem only holds for continuous approximation and describes the fitting of a polynomial of degree m-1 to a polynomial of next highest degree.

In our experiment, we were using only a discrete number of points (evenly spaced) and non-polynomial functions. However, our preliminary analyses have indicated that although the roots of the Chebyshev polynomial are not always the interpolation points defining the best L_1 solution, they are usually very "close". This result suggests that one might consider using such values x_k as starting points for discrete L_1 polynomial approximation algorithms. Further research in the area is presented in [8].

[1] A mth order Chebyshev polynomial of the second kind has the following form:

$$U_m(x) = \sum_{l=0}^{m/2} (-1)^l \frac{(1-m)!}{l!(m-2l)!} (2x)^{m-2l}$$

where $U_m(1)$ is defined to equal m+1.

VI. SUMMARY AND RECOMMENDATIONS

In summarizing the results of the work described here, several conclusions and recommendations can be made. First, the results clearly indicate that solution quality is substantially improved by a final (double-precision) reinversion of the optimal basis matrix. We recommend that this procedure be implemented to supplement any of the three single precision L_1 codes when solving polynomial approximation problems of the type described here.

In an earlier experiment [14] using L_1 problems with different structure, all four codes consistently produced correct solutions so that CPU time was the only distinguishing performance measure. Code II used significantly less CPU time than any of the other three codes. In particular, Code I, the double-precision code, was relatively slow in obtaining solutions. The results from the polynomial approximation experiment contrast sharply. Solution quality is not consistent among codes. In fact, based on the measures of solution quality presented in this report, it can be concluded that Code II is not well-suited for solving the type of polynomial approximation problems utilized in this experiment. On the other hand, Code I not only produced more correct solutions than any of the other three codes, but it utilized relatively little CPU time in doing so. It is conjectured that the use of double-precision arithmetic in Code I was costly in terms of increased CPU time in the first experiment, but paid off in the second experiment where the generated problems were numerically more difficult to solve.

Codes III and IV are fairly comparable in terms of solution quality. Recall, however, the eight problems for which Code IV utilized excessive CPU time. A user of Code IV is advised to impose a limit on execution times to avoid this occurrence. Furthermore, to insure replicability when several problems are solved sequentially by Code IV, the user is advised to reinitialize the random number generator between problems as discussed in Section II. We note in passing that the use of a double-precision variable in Code III to force the double-precision accumulation of inner products appears to be a cost-effective approach to improving solution quality.

Based on the observation that interpolation points are nearly identical for different functional forms, we anticipate that additional investigation of this phenomenon may well lead to a recommended "starting basis" for this type of problem. The theoretical explanation for the continuous case suggests the use of the roots of Chebyshev polynomials as a starting solution. Further study should be conducted to determine the practical implications of this suggestion in the discrete case.

We caution the reader that the results obtained from this experiment are valid only for the type of problem described herein. Further evaluation of code performance on polynomial approximation problems should address the situation in which it is desired to find the best polynomial approximation to observed data. This type of problem can be generated by POLY2 and should be utilized in future L_1 code comparison efforts. The importance of examining many classes of problems is emphasized by the difference in the results reported here and those in earlier experiments [14].

The evaluation described in this paper was conducted in accordance with a systematic methodology for obtaining statistically valid results. The set of test problems was randomly generated within a well-defined, controlled structure known to arise in real world applications. Performance measures were identified for both solution quality and computational efficiency. Finally, the conditions under which CPU time was measured were chosen to minimize variability. The results of the experiment, therefore, permit a valid ranking among the four codes with respect to code performance on the class of problems covered by this experiment. Although the rankings may vary in another computing environment, the experimental approach can be replicated to obtain equally valid results. The general type of methodology presented in this paper should serve as a guide for future code comparison experiments.

VII. REFERENCES

[1] Abdelmalek, N. N., An Efficient Method for the Discrete Linear L_1 Approximation Problem, Mathematics of Computation, 29, 844-850 (1975).

[2] Appa, G. and Smith, C., On L_1 and Chebyshev Estimation, Mathematical Programming, 5, 73-87 (1973).

[3] Armstrong, R. D. and Frome, E. L., A Comparison of Two Algorithms for Absolute Deviation Curve Fitting, Journal of the American Statistical Association, 71, No. 354, 328-330 (1976).

[4] Barrodale, I. and Roberts, F. D. K., An Improved Algorithm for Discrete L_1 Linear Approximation, SIAM Journal on Numerical Analysis, 10, No. 5, 839-848 (1973).

[5] Barrodale, I. and Roberts, F. D. K., Solution of an Overdetermined System of Equations in the L_1 Norm, Communications of the Association for Computing Machinery, 17, 319-320 (1974).

[6] Bartels, R. H., Conn, A. R., and Sinclair, J. W., Minimization Techniques for Piecewise Differentiable Functions: The L_1 Solution to an Overwhelmed Linear System, SIAM Journal on Numerical Analysis, 15, No. 2, 224-241 (1978).

[7] Davis, P. J., Interpolation and Approximation, (Blaisdell, New York, 1965).

[8] Domich, P. D. and Hoffman, K. L., A Near Optimal Starting Basis for Polynomial Approximation of Continuous Functions in the L_1-Norm, unpublished working paper, National Bureau of Standards.

[9] Domich, P., Lawrene, J., and Shier, D., Generators for Discrete Polynomial L_1 Approximation Problems, Journal of Research, National Bureau of Standards, 84, No. 6, November-December 1979.

[10] Dongarra, J. J., Moler, C. B., Bunch, J. R., and Stewart, G. W., LINPACK Users' Guide, Society for Industrial and Applied Mathematics, Philadelphia (1979).

[11] Forsythe, G. E., Generation and Use of Orthogonal Polynomials for Data-Fitting with a Digital Computer, SIAM Journal, 5, No. 2, 74-88 (1957).

[12] Gentle, J. E., Least Absolute Values Estimation: An Introduction, Communications in Statistics, B6, No. 4, 313-328 (1977).

[13] Gentle, J. E., Kennedy, W. J., and Sposito, V. A., On Least Absolute Values Estimation, Communications in Statistics, A6, No. 9, 839-845 (1977).

[14] Gilsinn, J., Hoffman, K., Jackson, R. H. F., Leyendecker, E., Saunders, P., and Shier D., Methodology and Analysis for Comparing Discrete Linear L_1 Approximation Codes, Communications in Statistics, B6, No. 4, 399-413 (1977).

[15] Hampel, F. R., A General Qualitative Definition of Robustness, Annals of Mathematical Statistics, 42, 1887-1896 (1971).

[16] Harter, H. L. Nonuniqueness of Least Absolute Values Regression, Communications in Statistics, A6, No. 9, 829-838 (1977).

[17] Hoffman, K. and Shier, D., A Test Problem Generator for Discrete Linear L_1 Approximation Problems, to appear in Transactions on Mathematical Software.

[18] Ortega, J. M., Numerical Analysis, A Second Course, Academic Press, New York (1972).

[19] Rice, J. R., The Approximation Functions, 1 (Addison-Wesley, Reading, Mass., 1964).

[20] Rice, J. R. and White, J. S., Norms for Smoothing and Estimation, SIAM Review, 6, No. 3, 243-256 (1964).

[21] Rivlin, T. J., An Introduction to the Approximation of Functions, (Blaisdell Publishing Company, Waltham, Mass., 1969).

[22] Shier, D. R., Neupauer, S. J., and Saunders, P. B., A Collection of Test Problems for Discrete Linear L_1 Data Fitting, National Bureau of Standards Internal Report NBSIR 79-1920, November 1979.

[23] Schlossmacher, E. J., An Iterative Technique for Absolute Deviations Curve Fitting, Journal of the American Statistical Association, 68, No. 344, 857-859 (1973).

[24] Spyropoulos, K., Kiountouzis, E. and Young, A., Discrete Approximation in the L_1 Norm, Computer Journal, 16, No. 2, 180-186 (1973).

[25] Timan, A. F., Theory of Approximation of Functions of a Real Variable, trans. by J. Berry (MacMillan Co., New York, New York, 1963).

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 86-3390	2. Performing Organ. Report No.	3. Publication Date MAY 1986

4. TITLE AND SUBTITLE

Evaluation of L_1 Codes Using Polynomial Approximation Problems

5. AUTHOR(S)

P.D. Domich, K.L. Hoffman, R.H.F. Jackson, P.B. Saunders, D.R. Shier.

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Center for Applied Mathematics
NBS

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

This paper presents the methodology and results of a computational experiment which compares the performance of four computer codes which determine the best discrete L_1 approximation to a continuous nonlinear function. The experiment utilizes 320 test problems created by a test problem generator. Several performance measures describe solution quality as well as computational effort.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

Algorithms testing; approximation; computational experiment; least absolute deviation; performance measures; polynomial approximation.

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

47

15. Price

$9.95

USCOMM-DC 6043-P80